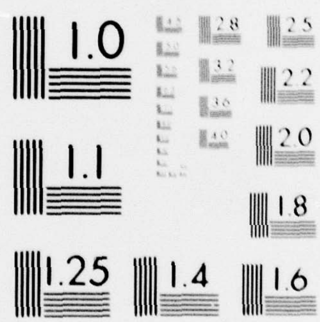
AD-A073 380 HONEYWELL SYSTEMS AND RESEARCH CENTER MINNEAPOLIS MN F/G 17/5
PROTOTYPE AUTOMATIC TARGET SCREENER.(U)
JUL 79 D E SOLAND, R C FITCH, D V SERREYN DAAK70-77-C-0248

UNCLASSIFIED 79SRC43 NL

| OF |

AD
A073380

END
DATE
FILMED
10-79
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

LEVEL II

PROTOTYPE AUTOMATIC TARGET SCREENER

by

D. E. Soland
R. C. Fitch
T. G. Kopet
D. V. Serreyn
R. C. Reitan
M. O. Schroeder

9 July 1979

QUARTERLY REPORT FOR PERIOD

1 April 1979 -- 30 June 1979

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

NIGHT VISION AND ELECTRO-OPTICS LABORATORY
FORT BELVOIR, VIRGINIA 22060

HONEYWELL
SYSTEMS & RESEARCH CENTER
2600 RIDGWAY PARKWAY
MINNEAPOLIS, MINNESOTA 55413

"The views, opinions, and/or findings contained in this report are those of the authors and should not be construed as an official department of the Army position, policy, or decision, unless so designated by other documentations."

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (WHEN DATA ENTERED)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOV'T ACCESSION NUMBER	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (AND SUBTITLE) PROTOTYPE AUTOMATIC TARGET SCREENER		5. TYPE OF REPORT/PERIOD COVERED Quarterly Progress Report 1 April - 30 June 1979
		6. PERFORMING ORG. REPORT NUMBER 79SRC43
7. AUTHOR(S) D. E. Soland, D. V. Serreyn, R. C. Reitan, R. C. Fitch, T. G. Kopet, M. O. Schroeder		8. CONTRACT OR GRANT NUMBER(S) DAAK70-77-C-0248
9. PERFORMING ORGANIZATIONS NAME/ADDRESS Honeywell Systems and Research Center, 2600 Ridgway Parkway, Minneapolis, MN 55413		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 1 E263710DK70 14 010CJ
11. CONTROLLING OFFICE NAME/ADDRESS Night Vision and Electro-Optics Laboratory, Fort Belvoir, Virginia 22060		12. REPORT DATE July 9, 1979
		13. NUMBER OF PAGES 54
14. MONITORING AGENCY NAME/ADDRESS (IF DIFFERENT FROM CONT. OFF.)		15. SECURITY CLASSIFICATION (OF THIS REPORT) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (OF THIS REPORT)

Approved for public release, distribution unlimited.

17. DISTRIBUTION STATEMENT (OF THE ABSTRACT ENTERED IN BLOCK 20, IF DIFFERENT FROM REPORT)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (CONTINUE ON REVERSE SIDE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)

Infrared	Target recognition	Image enhancement
FLIR	Pattern recognition	
Target cueing	Image processing	
Target screening	Real time	

20. ABSTRACT (CONTINUE ON REVERSE SIDE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)

This report is the seventh quarterly progress report for contract DAAK70-77-C-0248, Prototype Automatic Target Screener. The objective of the effort is to design an automatic target screener to be used with thermal imaging systems employing common module components.

HD-168 REV 11/74

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 55 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (WHEN DATA ENTERED)

402349

SECURITY CLASSIFICATION OF THIS PAGE (WHEN DATA ENTERED)

SECURITY CLASSIFICATION OF THIS PAGE (WHEN DATA ENTERED)

CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (concluded)

LIST OF TABLES

SECTION I

INTRODUCTION AND SUMMARY

This is the seventh quarterly technical progress report for contract number DAAK70-77-C-0248, Prototype Automatic Target Screener (PATS). The first two quarterly reports document the Phase I design study. The third quarterly report provides a description of the final target classifier design for the target data base currently available and the results of the hardware and CPU1 software system design tasks. The fourth through sixth reports and this one describe further the subsystem design details and status of the hardware fabrication, software coding, and hardware checkout. This report covers the period from 1 April to 30 June 1979.

The program objective is to produce a design for an automatic target screener. The screener will reduce the task loading on the thermal imager operator by detecting and recognizing a limited set of high-priority targets at ranges comparable to or greater than those for an unassisted observer. A second objective is to provide enhancement of the video presentation to the operator. The image enhancement includes: 1) automatic gain/brightness control to relieve the operator of the necessity to continually adjust the display gain and brightness controls and 2) DC restoration to eliminate artifacts resulting from ac coupling of the infrared (IR) detectors.

Image enhancement will also include local area gain and brightness control to enhance variations of contrast and compress the overall scene dynamic range to match that of the display. This circuitry has been completed, and

examples of its performance on videotaped thermal image data were included in the first quarterly report, along with the circuit description.

The DC restoration image enhancement circuit eliminates the streaking associated with loss of line-to-line correlation on the displayed image because of the ac coupling of the detector channels.

This report consists of four sections. Section II describes the hardware status and design. Section III presents the software status and Section IV provides the plans for next reporting period.

SECTION II

HARDWARE DESIGN AND STATUS

This section describes the interval design, hardware status, preliminary results of edge and bright on actual imagery, power supplies chosen, and additional comments on the interface between CPU1 and CPU2. The implementation of the PATS hardware was described in previous reports. The general system hardware specification was covered in the third quarterly report. This was the target screening function only and not the image enhancement. The hardware, however, does include image enhancement functions.

The target screener with image enhancement was broken down into subparts which have been covered in various quarterly reports 1-7.

[1] D. E. Soland, et al., "Prototype Automatic Target Screener," Quarterly Progress Report No. 1, Contract DAAK70-77-0248, Honeywell Systems and Research Center, Minneapolis, Minnesota, January 15, 1978.

[2] D. E. Soland, et al., "Prototype Automatic Target Screener," Quarterly Progress Report No. 2, Contract DAAK70-77-0248, Honeywell Systems and Research Center, Minneapolis, Minnesota, June 15, 1978.

[3] D. E. Soland, et al., "Prototype Automatic Target Screener," Quarterly Progress Report No. 3, Contract DAAK70-77-0248, Honeywell Systems and Research Center, Minneapolis, Minnesota, September 1, 1978.

[4] D. E. Soland, Et al., "Prototype Automatic Target Screener," Quarterly Progress Report No. 4, Contract DAAK70-77-0248, Honeywell Systems and Research Center, Minneapolis, Minnesota, October 20, 1978.

[5] D. E. Soland, et al., "Prototype Automatic Target Screener," Quarterly Progress Report No. 5, Contract DAAK70-77-0248, Honeywell Systems and Research Center, Minneapolis, Minnesota, January 8, 1979.

[6] D. E. Soland, et al., "Prototype Automatic Target Screener," Quarterly Progress Report No. 6, Contract DAAK70-77-0248, Honeywell Systems and Research Center, Minneapolis, Minnesota, April 6, 1979.

[7] D. E. Soland, et al., "Prototype Automatic Target Screener," Quarterly Progress Report No. 7, Contract DAAK70-77-0248, Honeywell Systems and Research Center, Minneapolis, Minnesota, July 9, 1979.

The subparts which have been covered may be found in the following quarterly reports:

Subpart	Quarterly Report
Image Enhancement	2, 4
Edge Signal	3
Bright Signal	4
Interval Generation	7
CPU1	4
(Writable Control Store)	5
(CPU1/CPU2 Interface)	5, 6, 7
Memory 2 (Intensity Information)	4
CPU2	3
Symbol Generation	6
Sync and Timing	3, 5

As noted above, this section gives the interval design and additional information on the interface between the two CPU's. The hardware design tasks are complete except for some minor changes that might take place during checkout.

STATUS OF MODULES

Table 1 updates the status of the various modules to date. The percentage given is a rough estimate of progress for each task. The status of each functional subassembly is as follows:

- Image Enhancement--adaptive contrast enhancement board checked out for PATS board. DC Restore and global gain/bias will be checked out during integration phase.
- Edge Board--some modifications were made due to some wiring errors which affect threshold calculation. The board is checked out and functional. An example of raw video and edge output is shown in Figure 1 for some FLIR imagery.
- Bright Board--error found in background filter implementation; CCD's fine tuned for optimum bandwidth. An example of bright (hot) output is shown in Figure 2.
- Interval--design completed, two boards required for implementation, each board having about 50 IC's. One board built with second about half done. Checkout to begin soon. A description of the design is included in this report.
- CPU1--six boards built, including sequencer, ALU with multiplier, Memory #1, Memory #1 constants, DMA/FIFO interface, and CPU1/CPU2 interface sequencer checked out using writable control store. No problems encountered; prom program storage in operational system is a potential problem.

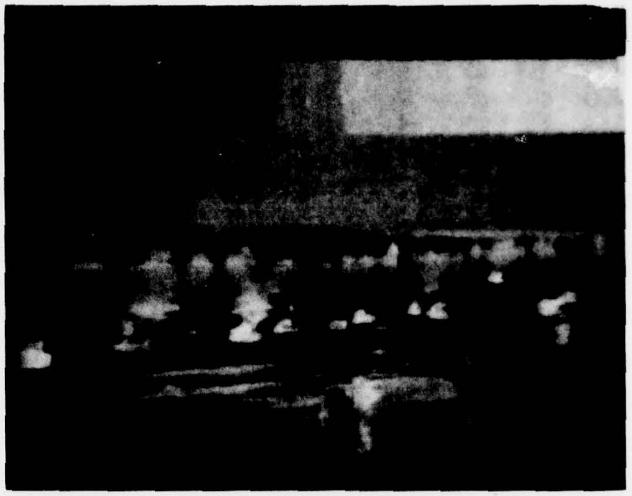


Figure 1A. FLIR Raw Video

Figure 1B. "Edge" Output for 1A

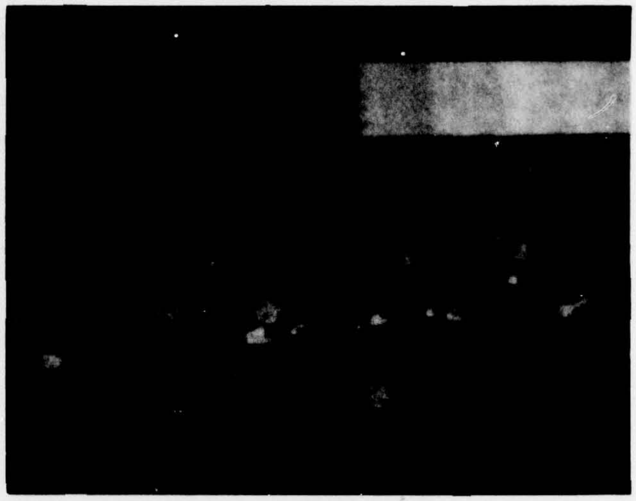


Figure 2A. FLIR Raw Video

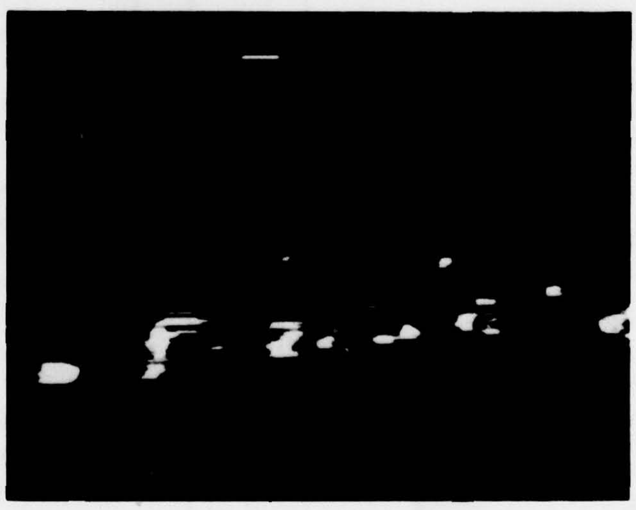


Figure 2B. Bright (HOT) Output for 2A

TABLE 1. STATUS OF PATS HARDWARE (PERCENTAGE COMPLETED)

Subpart	Boards	Design	Schematics	Build	Checkout Excluding Integration
Image Enhancement					
Adaptive Contrast Enhancement	1	100	100	100	100
DC Restore	1	100	100	100	0
Edge	1	100	100	100	95
Bright	1	100	100	100	95
Interval	1	100	100	75	0
CPU1 (Digital Processing Subsystem)					
Processor Inc Multiplier	1	100	100	100	100
Microprogram Memory Sequencer	1	100	100	100	100
FIFO/DMA I/F	1	100	95	95	50
Memory 1	2	100	100	100	100
CPU1/CPU2 I/F (inc in CPU2)	1	NA	NA	NA	90
Memory 2					
A/D, Summation	1	100	100	100	100
Memory Control and Refresh	1	100	100	100	100
Memory 512 x 512 x 2	4	100	100	100	100
CPU2					
CPU with 16K Memory (KD 11-HC)	2	NA	NA	NA	100
PROM Board (MRV11-AA)	1	NA	NA	0	0
Serial Port (DLV11) inc Printer/ Keyboard	1	NA	NA	NA	100
Refresh/Bootstrap (REV11-C)	1	NA	NA	NA	100
Floppy Controller (RXV11-BA) inc Floppies	1	NA	NA	NA	100
Symbol Generator (one included in CPU2)	2	100	100	100	90
Sync and Timing					
Sync Separator and Video Switches	1	100	100	100	100
Sync Generation	1	100	90	100	100
Writable Control Store	1	100	50	100	100

NA-Not applicable to PATS Design Tasks

ALU with Multiplier--checked out okay, individual control proms burned for operation. Revision at gate level taking place during checkout for timing considerations.

Memory #1--all 16K x 16-bit chips in and working. Extensive memory test remains to be done.

Memory #1 Constants--prom storage of constants needs to be worked on. Presently using RAM for storage of constants.

DMA/FIFO--checked out except for FIFO operation.

CPU1/CPU2 interfaces--additional chips added for interface; description included as part of this report.

- Memory #2--total of six boards built and initial checkout complete. This includes memory planes.
- CPU2--this computer was purchased; all parts have been assembled and are functioning. This computer may be removed from operational system and used in training and evaluation only.
- Symbol Generator--build and checkout completed for two boards: one board fits into CPU2 and will be rebuilt in cards used for most of PATS hardware, second board does vector generator. An example of output from symbol generator is shown in Figure 3.
- Sync and Timing--both boards built and checked out; work well with 525-line camera; PROM will have to be blown for 875-line system.

Figure 3A. FLIR Raw Video

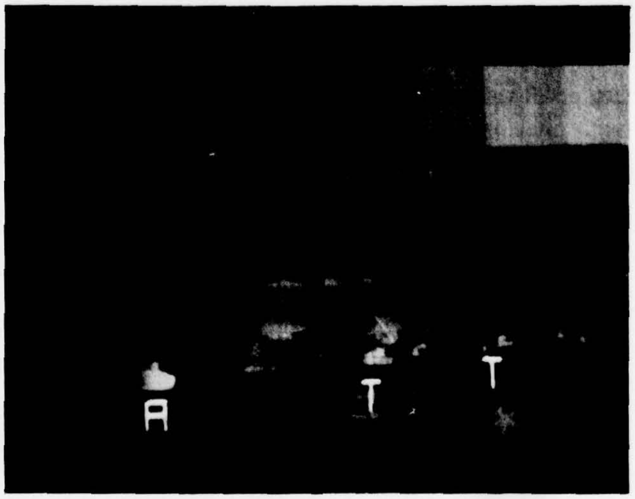


Figure 3B. Symbology Example

- Writable Control Store--used as part of MDS system for microcode development and checkout; working and being used as part of CPU1 checkout.

POWER SUPPLY

The power required for the system was given in the last quarterly report. These requirements are:

Supply Voltage	Current
+5	24.6
+12	2.5
+15	1.6
-15	2.0
-5	.5

These values exclude CPU2 power except for the symbol generator board. A total of four power supplies have been ordered for the system. The ±5 volt power supply has been ordered from Arnold Magnetics. Each side has current capability of 2.5 Amps. Two +5V, 20A supplies plus one +12V 8.3A power supply has been ordered from KEC.

All the power supplies will handle 47-440 Hz input and can be used both in the lab and in an aircraft.

INTERVAL DESIGN

The design of the PATS interval circuits includes the implementation in bipolar-TTL logic a number of PAT functions:

- Generation of an interval based on past, present, and future edge, hot and cold signals
- Validation of an interval as meeting certain practical constraints
- Storage and generation of key interval-related data; intensity sums, bright, width and interval counts, and background estimates
- Making interval data available to the processor memory and instructing it that valid data is ready.

The implementation of these functions is described in the subsections which follow. See Figure 4.

INTERVAL GENERATION

The generation of an interval is based on the bidirectional interval extraction algorithm shown in Figure 5. Note that the algorithm is to be applied on the pixels of the line from left to right and right to left. For simulation purposes this is easy, since the hot, cold and edge data is available in memory. However, for real-time applications, the data is available only in left to right or time increasing format. Hence it is necessary to be able to perform a set of alternate tests on the left to right data that would yield the same results as if operating on the line from right-to-left. That is, the new tests should turn on an interval where we would have turned one off in the right to left sense. Inverting the appropriate tests of Figure 5, we arrive at the modified turn on/off criteria shown in Figure 6. This new algorithm properly generates intervals when the data is available in only one "direction."

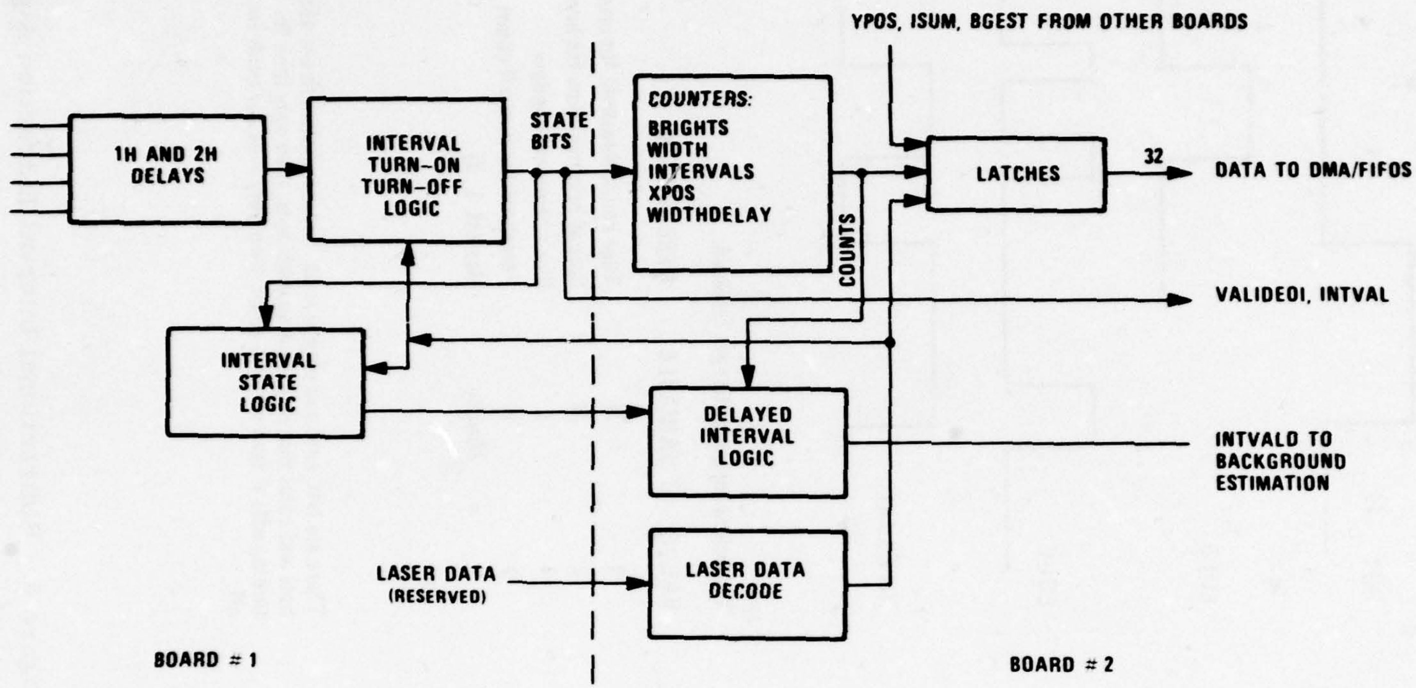
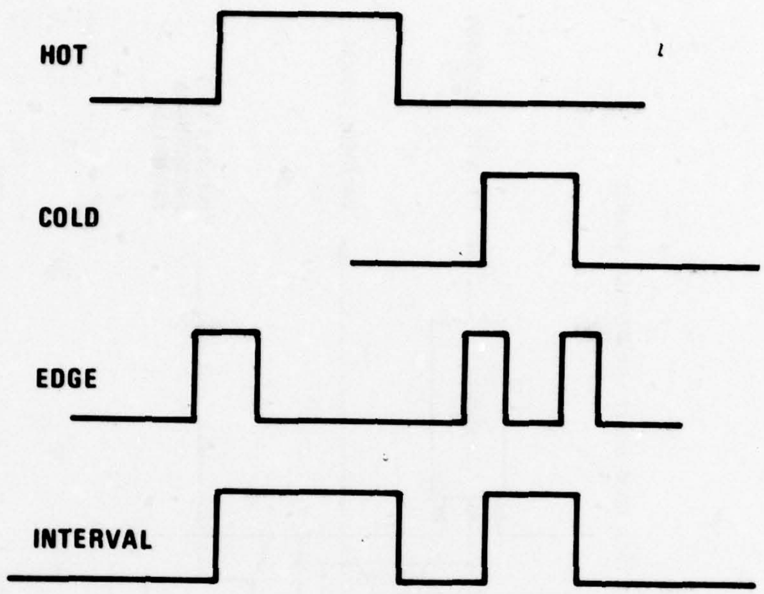


Figure 4. Interval Boards



The following quantities are involved:

VALUE	VARIABLE	DESCRIPTION
3	N	Edge count threshold for interval turn on
2	M	Size of brightwindow for interval turn on
4	L	Size of turn off window
3	K	Threshold for turn off count
4	Margin	Max (M, L, 3)

There are hot, cold and edge signals. Separate intervals are obtained for hots and colds that is, if turn on with hots, then only look for hots for turning off; if turn on with colds, then only look for colds for turning off.

Figure 5. Bidirectional Interval Extraction Algorithm

Note: The algorithm below is applied to each scan line forwards (left to right) and backwards (right to left).

Let j be the jth position along the ith scan line and:

$I_{i,j}$	=	1 or 0	FOR INTERVAL OR NO INTERVAL
$H_{i,j}$	=	1 or 0	FOR HOT OR NOT HOT
$C_{i,j}$	=	1 or 0	FOR COLD OR NOT COLD
$E_{i,j}$	=	1 or 0	FOR EDGE OR NO EDGE

Always: $j = 1,2 \ldots, (NC-MARGIN + 1)$
(nc = no. of columns (total no. of pixels along scan line)

Hot interval algorithm (likewise use $C_{i,j}$ for cold intervals)

SET $j = 1$

5: IF $\sum_{p=0}^{M-1} H_{i,j+p} = M$ GO TO 10

ELSE $I_{i,j} = 0,\ j \leftarrow j+1$, GO TO 5

10: IF $\sum_{p=-2}^{2} E_{i-1,\,j+p} \geqslant N$ OR

$\sum_{p=-2}^{2} E_{i,\,j+p} \geqslant N$ OR

$\sum_{p=-2}^{2} E_{i+1,\,j+p} \geqslant N$ THEN GO TO 20

ELSE IF $\sum_{p=-1}^{1} E_{i-1,\,j+p} > 0$ AND

$\sum_{p=-1}^{1} E_{i,\,j+p} > 0$ AND

$\sum E_{i+1,\,j+p} > 0$ THEN GO TO 20 ELSE: $j \quad j+1$ GO TO 5

20: Turn on interval (here hot)

21: IF: $H_{i,j} + H_{i,j+1} + \ldots H_{i,j+L-1} < K$

Then: interval is turned off, go to 5 Else: interval is on, $I_{i,j} = 1,\ j \leftarrow j+1$, GO TO 21

Figure 5. Bidirectional Interval Extraction Algorithm (concluded)

- **TURN ON AN INTERVAL IF (INTON)**

$$\sum_{p=0}^{L-1} H_{i,j-p} \geq K \qquad \text{(HRLON)}$$

OR

$$\sum_{p=0}^{L-1} C_{i,j-p} \geq K \qquad \text{(CRLON)}$$

OR

$$\sum_{p=0}^{M-1} H_{i,j+p} = M \qquad \text{AND FILEDGE TRUE} \qquad \text{(HLRON)}$$

OR

$$\sum_{p=0}^{M-1} C_{i,j+p} = M \qquad \text{AND FILEDGE TRUE} \qquad \text{(CLRON)}$$

"(NAME)" IS LOGICAL SIGNAL NAME FOR TRUE RESULT

- **TURN INTERVAL OFF IF (INTOFF)**

$$\sum_{p=0}^{L-1} H_{i,j+p} < K \qquad \text{(HLROFF)}$$

OR

$$\sum_{p=0}^{L-1} C_{i,j+p} < K \qquad \text{(CLROFF)}$$

OR

$$\sum_{p=0}^{L-1} H_{i,j-p} < K \qquad \text{(HRL1OFF)}$$

OR

$$\sum_{p=0}^{L-1} C_{i,j-p} < K \qquad \text{(CRL1OFF)}$$

OR

$$\sum_{p=0}^{M-1} H_{i,j-p} = M \qquad \text{AND } H_{i,j+1} = 0 \qquad \text{AND FILEDGE (CHRL2OFF)}$$

OR

$$\sum_{p=0}^{M-1} C_{i,j-p} = M \qquad \text{AND } C_{i,j+1} = 0 \qquad \text{AND FILEDGE (CRL2OFF)}$$

Figure 6. Undirectional Interval Extraction

IN SUMMARY:

- **TURN ON IF**

  HRLON or CRLON or HLRON or CLRON

- **TURN OFF IF**

  HLROFF or CLROFF or HRL1OFF or CRL1OFF or HRT2OFF or CRL2OFF

FILEDGE (used to indicate occurrence of edge at start or end) is true if a or b is true below.

a. $$\sum_{p=-2}^{2} E_{i-1,\, j+p} \geqslant N \quad \text{OR}$$

$$\sum_{p=-2}^{2} E_{i,\, j+p} \geqslant N \quad \text{OR}$$

$$\sum_{p=-2}^{2} E_{i+1,\, j+p} \geqslant N$$

b. $$\sum_{p=-1}^{1} E_{i-1,\, j+p} > 0 \quad \text{AND}$$

$$\sum_{p=-1}^{1} E_{i,\, j+p} > 0 \quad \text{AND}$$

$$\sum_{p=-1}^{1} E_{i+1,\, j+p} > 0$$

Figure 6. Undirectional Interval Extraction (concluded)

A block diagram of the generation logic is shown in Figure 7. The edge signals at the previous line and next line are obtained via cascaded 256-bit digital shift registers to serve as 1H delays. To delay equalize the hot and cold signals, these are each delayed by 2H in a similar fashion. To obtain the required pixel delays along a line, eight-bit shift registers are used. Thus, we have available all the Hi, j, Ci, j and Ei, j as specified by Figure 6. To perform the sums

$$\sum_{p=0}^{-(L-1)} H_{i,j+p}, \quad \sum_{p=0}^{L-1} H_{i,j+p}, \quad \sum_{p=0}^{-(L-1)} C_{i,j+p}, \text{ and } \sum_{p=0}^{L-1} C_{i,j+p}$$

Eight-bit by 32-word PROMS are to be used to provide a number of combinational output functions, as shown in Table 2. The prom outputs, used directly or compared against a preset digital value, allow the performance of all the tests outlined by the algorithm. The test for M consecutive Hi, j, or Ci, j being 1, is implemented using AND gates to look at M adjacent pixels along a line. Note that N, L, M, and K are all adjustable from one to five.

All of the potential turn on/off logical signals are next "OR" ed together to provide a common turn on signal, and a single "color" turn off signal. That is, the turn off criterion used must be for Hi, j if a hot interval is started and only Ci, j if a cold interval is started.

The generation logic up to this point is strictly combinational. That is, the turn on/off signals can change with each new pixel enetered. Now state-logic is needed to do the actual interval generation. In particular, seven J-K flip flops are used to provide memory and sequencing of the interval generation algorithm. The outputs from each flip flop are discussed below.

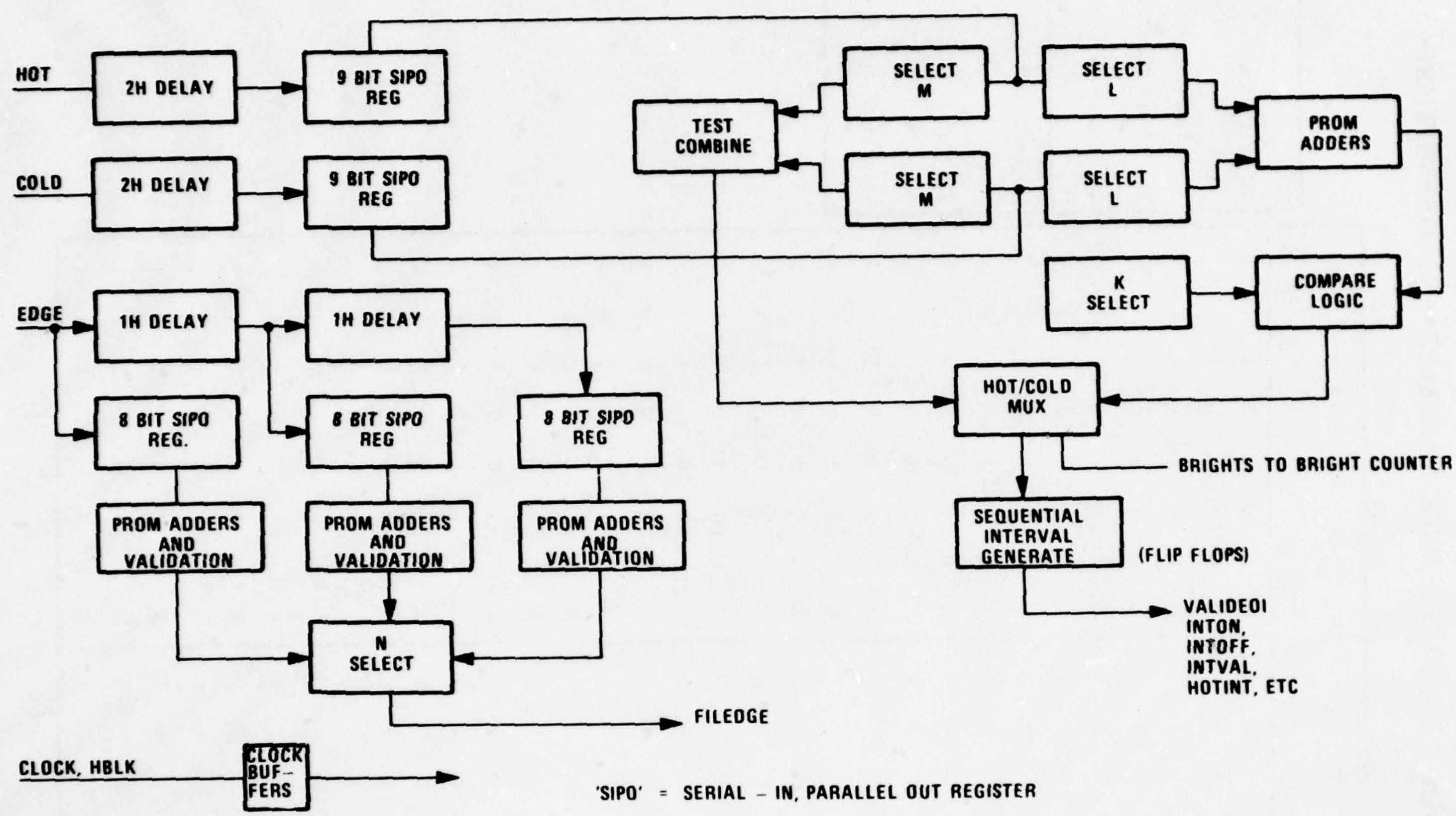


Figure 7. Interval Generation

TABLE 2. PROM PROGRAM MAP AND DEFINITIONS

Address (A_i)					Data Bits							
A_4	A_3	A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	X	0	0	0	0	0	0	0
0	0	0	0	1	X	0	0	1	0	0	0	1
0	0	0	1	0	X	0	0	1	0	0	0	1
0	0	0	1	1	X	0	1	0	0	0	1	1
0	0	1	0	0	X	0	0	1	0	0	0	1
0	0	1	0	1	X	0	1	0	0	0	1	1
0	0	1	1	0	X	0	1	0	0	0	1	1
0	0	1	1	1	X	0	1	1	0	1	1	1
0	1	0	0	0	X	0	0	1	0	0	0	1
0	1	0	0	1	X	0	1	0	0	0	1	1
0	1	0	1	0	X	0	1	0	0	0	1	1
0	1	0	1	1	X	0	1	1	0	1	1	1
0	1	1	0	0	X	0	1	0	0	0	1	1
0	1	1	0	1	X	0	1	1	0	1	1	1
0	1	1	1	0	X	0	1	1	0	1	1	1
0	1	1	1	1	X	1	0	0	1	1	1	1
1	0	0	0	0	X	0	0	1	0	0	0	1
1	0	0	0	1	X	0	1	0	0	0	1	1
1	0	0	1	0	X	0	1	0	0	0	1	1
1	0	0	1	1	X	0	1	1	0	1	1	1
1	0	1	0	0	X	0	1	0	0	0	1	1
1	0	1	0	1	X	0	1	1	0	1	1	1
1	0	1	1	0	X	0	1	1	0	1	1	1
1	0	1	1	1	X	1	0	0	1	1	1	1
1	1	0	0	0	X	0	1	0	0	0	1	1
1	1	0	0	1	X	0	1	1	0	1	1	1
1	1	0	1	0	X	0	1	1	0	1	1	1
1	1	0	1	1	X	1	0	0	1	1	1	1
1	1	1	0	0	X	0	1	1	0	1	1	1
1	1	1	0	1	X	1	0	0	1	1	1	1
1	1	1	1	0	X	1	0	0	1	1	1	1
1	1	1	1	1	X	1	0	1	1	1	1	1

$(D_4, D_5, D_6) = \sum_{i=0}^{4} A_i$

D_4 = Least significant bit

D_6 = Most significant bit

D_0 true if $\sum_{i=0}^{4} A_i \geq 1$

D_1 true if $\sum_{i=0}^{4} A_i \geq 2$

D_2 true if $\sum_{i=0}^{4} A_i \geq 3$

D_3 true if $\sum_{i=0}^{4} A_i \geq 4$

1. INTON--this signal becomes true on the next clock cycle after an interval turn-on criterion has been met. It will remain true until two clock cycles after a turn-off criterion has been met.

2. INTOFF--will become true for one clock cycle on the first clock cycle following the occurrence of a turn-off criterion. INTOFF cannot become true unless INTON is already true.

3. ES (edge at start)--will become true on the next clock cycle following the occurrence of interval turn-on if the turn on was initiated by the coincidence of hot or cold and edge signals.

4. EE (edge at end)--like ES except that it follows the turn-off event time if caused by an edge-turn-off.

5. HOTINT--this signal indicates the "color" (hot or cold) of an interval. If true, the interval is hot; cold if false. It can become true if the turn-on criterion was due to a hot signal. HOTINT is reset after interval turn-off.

6. INTVAL--this signal is exactly like INTON except that it becomes true <u>during</u> the clock cycle in which interval turn on occurs. This signal is the interval level signal used by other PATS boards.

7. VALIDEOI (valid end of interval)--this signal will be true high for one clock cycle following the occurrence of INTOFF if the interval is declared valid. This pulse is used to signal the DMA board that a new set of interval data has been latched and should be read-in. The validation of the interval is discussed next.

INTERVAL VALIDATION

At present, an interval is declared valid if all of the following are true:

- Either EE or ES is true (an edge occurred within the interval).
- The width of the interval after turn-off is $\leq 63_{10}$ pixels.
- There have been fewer than 21_{10} intervals declared valid on the current scan line.

These tests are "AND"ed together to allow the occurrence of VALIDEOI. The result of the AND is called FIFOLOAD and is used to control the loading of a 16-bit x 64-word FIFO to provide a 1H delayed replica of the interval. FIFOLOAD is also used to latch all relevant interval data for use in the DMA board. This section is discussed in more detail below.

INTERVAL DATA STORAGE

Associated with each valid interval are a number of first-level features Each of these features is latched at the end of an interval into octal tristateable latches. The data put into these latches consists of (see Figure 8):

• YPOS00 - YPOS09	10 bits indicating the scan line number
• INCT0 - INCT4	5 bits: the number of intervals on the line
• INTV21	1 bit indicating that ≥ 21 intervals occurred on the present scan line
• HOTINT	1 bit indicating the interval temperature
• WCO - WC4	5 bits indicating the width of an interval from 0 to 31

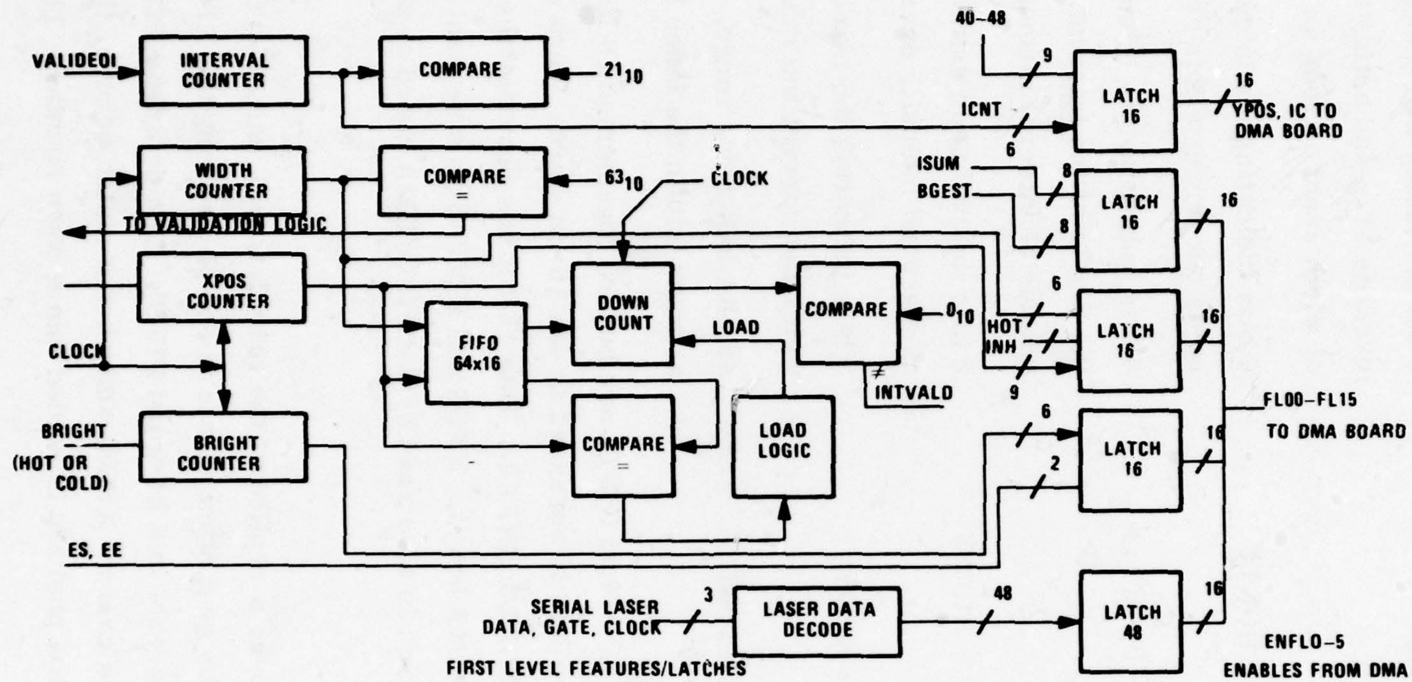


Figure 8. Interval Board #2

•	WIDTHBAD	1 bit indicating that the interval width exceeds 31_{10} and acting as the 2^5 bit of width count as long as $0 \le \text{width} \le 63$
•	BCNT0 - BCNT5	6 bits indicating the number of hots or colds occurring within an interval
•	ISUM0 - ISUM7	8 bits indicating the intensity sum on the pixels of the interval. Derived outside of interval circuits
•	BGEST0 - BGEST7	8 bits indicating an estimate of the background intensity over an interval
•	XPOS0 - XPOS8	9 bits indicating the pixel number at which the current interval started
•	RANGE, RANGER, AZIM (Reserved)	48 bits indicating range, velocity and azimuth within the laser FOV

All data, except YPOS and ICNT, are bussed together on 16 lines called FL00-FL15. The data is accessed as six 16-bit words, each enabled on the tri-state bus by ENFL0-ENFL5. This 16-bit bus provides the input to the first level FIFO/DMA board. YPOS and ICNT form a separate 16-bit, non tri-state word which is also read by the FIFO/DMA board at the end of every scan line.

As mentioned above, a replica of the valid interval, but delayed by one line time, must also be generated. This is done by storing the XPOS and WC data for valid intervals in a 64-word FIFO). The delayed XPOS data is then compared with the current XPOS, and when a match occurs, the width count associated with that position is loaded into a down counter. The 5-bit output

of the down counter is then compared against zero, and the condition that the count is greater than zero is used as the delayed interval signal. This scheme was chosen over digital delay lines since it obviates the need to make a look-ahead decision about the validation of an interval, as well as the more complex clocking scheme that the delay lines would require. An example of the interval generation timing is in Figure 9.

CPU1 - CPU2 INTERFACE

An interface between CPU1 and the CPU2 DMA board has been designed, built, and will be checked out shortly. The CPU2 DMA is the Computer Technology DMA-L11 which was described in the previous quarterly report.[1] The DMA-L11 together with the interface between it and CPU1 form the complete CPU1 - CPU2 interface.

The block diagram of this interface appears in Figure 10. All I/O transfers on the CPU1 side of the interface are accomplished via programmed I/O; all transfers on the CPU2 side are accomplished via DMA. The speed of the interface is limited by the speed of the CPU2 DMA which is approximately 800 kHz. The interface protocols are described in the two sections following. Note that all CPU2 addresses are in octal and all CPU1 addresses are in hex.

CPU1 TO CPU2 DATA TRANSFER

CPU1 to CPU2 data transfers always have a fixed length of 102 words and always go into a fixed buffer in CPU2. To accomplish the transfer, the following must occur:

[1] D. E. Soland, et al., "Prototype Automatic Target Screener," Quarterly Progress Report No. 6, Contract No. DAAK70-77-C-0248, Honeywell Systems and Research Center, Minneapolis, MN, April 6, 1979, pp. 50, 53-65.

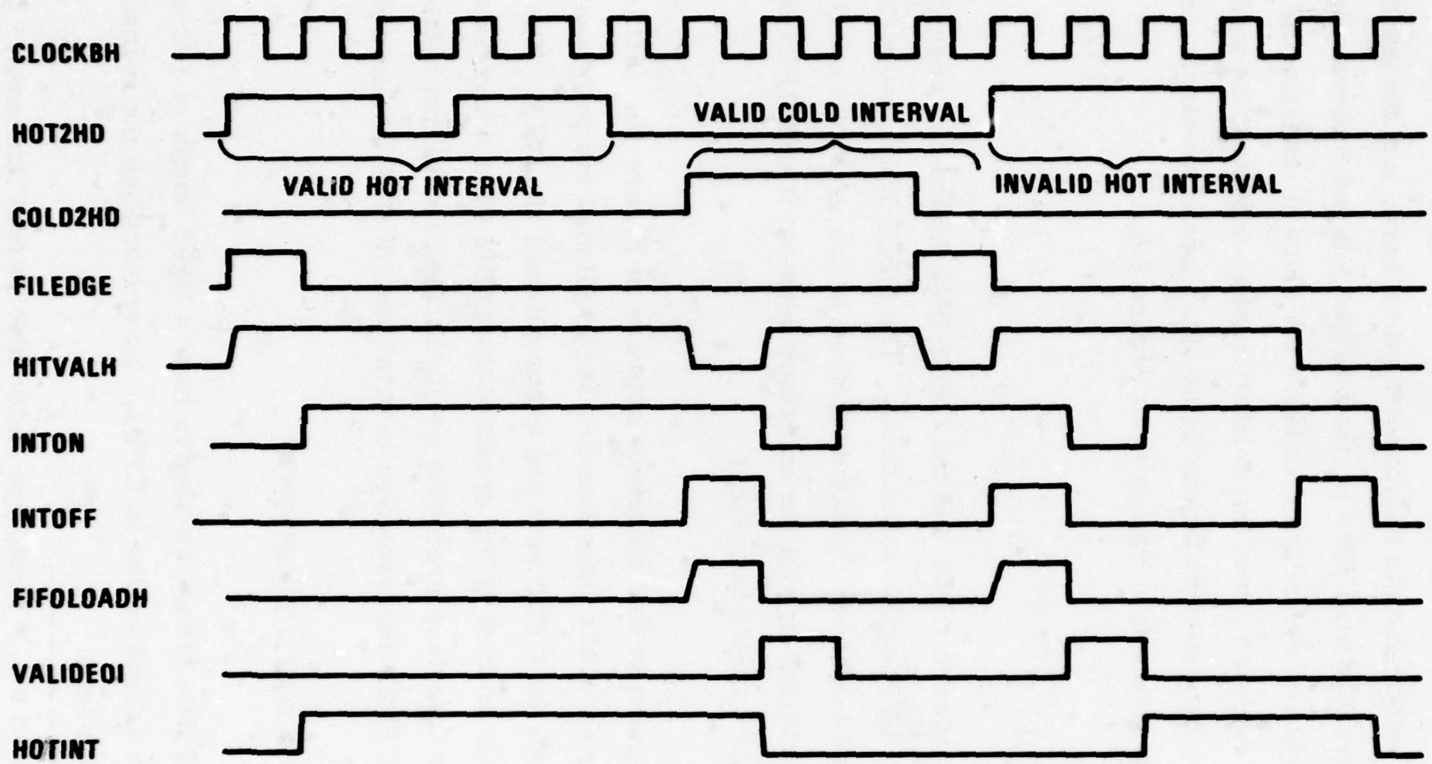


Figure 9. Example of Interval Timing

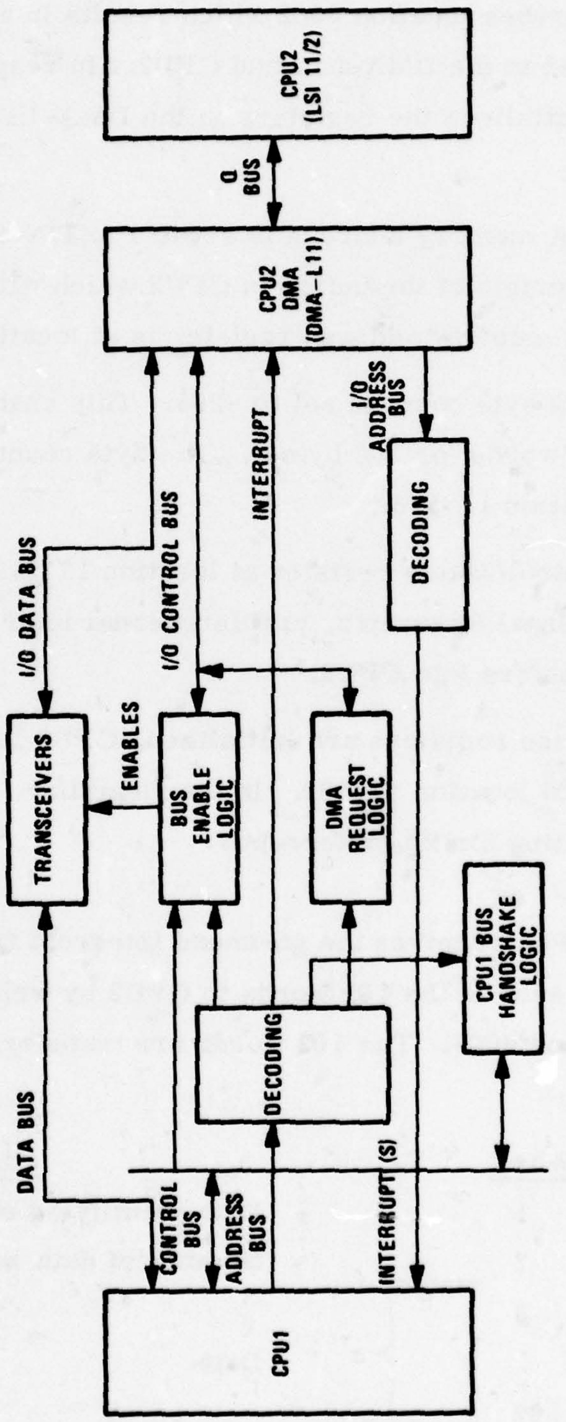


Figure 10. Block Diagram of CPU1-CPU2 Interface

1. CPU1 strobes location 6002 which results in an interuupt being generated to the DMA-L11 and CPU2. In response to this interrupt, CPU2 initializes the registers in the DMA-L11 in the following manner:

- DMA memory address is set to PASTIN-2, where PASTIN is the origin of the buffer in CPU2 which will receive the data. The memory address register is at location 177514.
- DMA byte count is set to -205. This enables a transfer of 102 words (or 204 bytes). The byte count register is at location 177516.
- Control/status register at location 177512 is written to disable external interrupts, enable internal interrupts, and enable transfers into CPU2.

After these registers are initialized, CPU2 interrupts CPU1 by writing to location 177500. In the meantime, CPU1 has been in a loop waiting for this interrupt.

2. When CPU1 receives the go-ahead interrupt from CPU2, it proceeds to DMA each of the 102 words to CPU2 by writing each one in turn to location 6003. The 102 words are transferred in the following format:

Word	Contents
1	Data identifying code
2	Number of data words being transferred
3 ⋮ 102	Data

Word 1 tells CPU2 the type of data being transferred (for example, feature data, debugging or error messages, etc.). Word 2 gives the total number of data words which will ultimately be transferred to CPU2. If this number is greater than 100, CPU2 will know that it can expect more data in a subsequent DMA transfer, and that this data will logically belong with the data from the current transfer. For example, if CPU1 wishes to send CPU2 a total of 250 words, it will, in turn, DMA each of the following three records to CPU2:

Record 1: word 1 = data ID code
word 2 = 250
word 3 - word 102 = data

Record 2: word 1 = data ID code
word 2 = 150
word 3 - word 102 = data

Record 3: word 1 = data ID code
word 2 = 50
word 3 - word 52 = data
word 53 - word 102 = don't care

If on the other hand, CPU1 wishes to transfer only 73 words to CPU2, it will DMA one record with the following contents:

word 1 = data ID code
word 2 = 73
word 3 - word 75 = data
word 76 - word 102 = don't cares

3. When all the data from a single 102-word record has been transferred, byte count overflow in the DMA-L11 will cause it to interrupt CPU2. CPU2 will then remove the data from the buffer and place it elsewhere in memory. Finally, CPU2 will rewrite the control status register to enable external interrupts and disable internal interrupts.

CPU2 TO CPU1 DATA TRANSFER

Unlike CPU1 to CPU2 transfers, these transfers are not restricted to any particular record size. CPU2 can set the word count register on the DMA-L11 to any number provided it does not overflow the word count register or the memory size in CPU1. To transfer the data, the following must occur:

1. CPU2 interrupts CPU1 by writing to location 17500. Prior to doing this, CPU2 set up all the registers in the DMA-L11 appropriately. Internal interrupts on the DMA-L11 are enabled.

2. CPU1 responds to the interrupt by DMAing three words of data from CPU2. The three words are as follows:

Word	Contents
1	Data identifying and/or command code
2	Starting memory address
3	Number of data words being transferred

Word 1 is used to indicate into which CPU1 memory data is to be placed (that is, either Memory 1 or Memory 2). It is also used to pass commands which direct CPU1 processing. If after examining word 1, CPU1 determines that a data transfer is being requested

by CPU2, CPU1 sets itself up to handle the transfer using the additional data in words 2 and 3. Word 2 is the starting address at which data will be placed in the selected memory. For this purpose, Memory 2 is mapped into a one-dimensional address space which spans each row from left to right, starting with the left-most pixel of the first row.

Word 3 is the total number of data words CPU1 will be transferring from CPU2. CPU1 DMAs data from the DMA-L11 by reading location 6003. A read must be done for every word being transferred.

3. When all the transfers have been made, byte count overflow on the DMA-L11 causes an interrupt of CPU2. CPU2 then enables external interrupts on the DMA-L11 to enable an interrupts from CPU1.

SUMMARY

It is the nature of the DMA-L11 that all DMA transfers, whether to or from CPU2, must be requested a word at a time by the external device, that is, CPU1. The protocol described meets this requirement by using interrupts to signal between CPU1 and CPU2. Data from CPU1 to CPU2 is always transferred in fixed length records, one record at a time, to a fixed buffer in CPU2. This avoids the necessity of passing a starting address and word count from CPU1 to CPU2. Transfers from CPU2 to CPU1 do not have this restriction. CPU2 is notified of the completion of a DMA transfer by an interrupt from the DMA-L11. The DMA-L11 will also interrupt CPU2 in

the event of DC power failure or a bus timeout error on the LSI11 Qbus. If either of these events occurs in the course of a DMA transfer, CPU2 aborts further transfers by interrupting CPU1 with a write to 177502. CPU1 responds to this interrupt by ceasing DMA requests.

SECTION III

SOFTWARE STATUS

This section presents information on the status of the microcode software development and the CPU2 software. Even though CPU2 may be eliminated as part of the flight test system, certain software is being developed relative to testing of CPU1, interframe analysis and symbol generation. This software will have to be transferred to CPU1 microcode when CPU2 is eliminated.

CPU2 SOFTWARE

The quarterly report dated 8 January 1979 contains a description of all CPU2 software Tables 8 and 9 from that report are reproduced here, for easy reference, as Tables 3 and 4. In Tables 3 and 4, the routines with the * next to them have not been tested. All others have been coded and tested. Testing was performed with artificial data designed to allow tracing through control structures as well as testing computations.

This testing has been done for major divisions of the software. For example, in Table 4 all the routines listed under the heading "interframe analysis" have been tested together. But they were tested with stubs for APC, UIT, etc.. The stubs just printed a message rather than doing anything. Similarly, the symbol generation routines have been tested using DISPTST to call them, but the interframe analysis routines were replaced with stubs.

With this procedure, major portions of the hierarchy chart (Figure 11) have been tested independently. Now they can be tested together.

TABLE 3. CPU2 SOFTWARE USED PRIMARILY FOR DIAGNOSTICS

CHEKMEM*
DISPTST
MEMTEST*
PROTOCL*
SIMULAT*

TABLE 4. OTHER CPU2 SOFTWARE

General	Interframe Analysis	Cueing
CPU1INPUT*	COSTIT	APC
DMACODE*	ELECT	CUEIT
DMAINVL*	GETUM	ERASE
DMARES1*	HUH	NONTAR
DMARES2*	INTER	NOTHNG
DMAVEC1*	PAIRUM	TANK
DAMVEC2*	PICK	TRUCK
INITIAL	DUMPSTAT	UIT
ISR1*		AACAN
ISR2*		AALNCH
OPINPUT		
SUPER		
TRAIN		

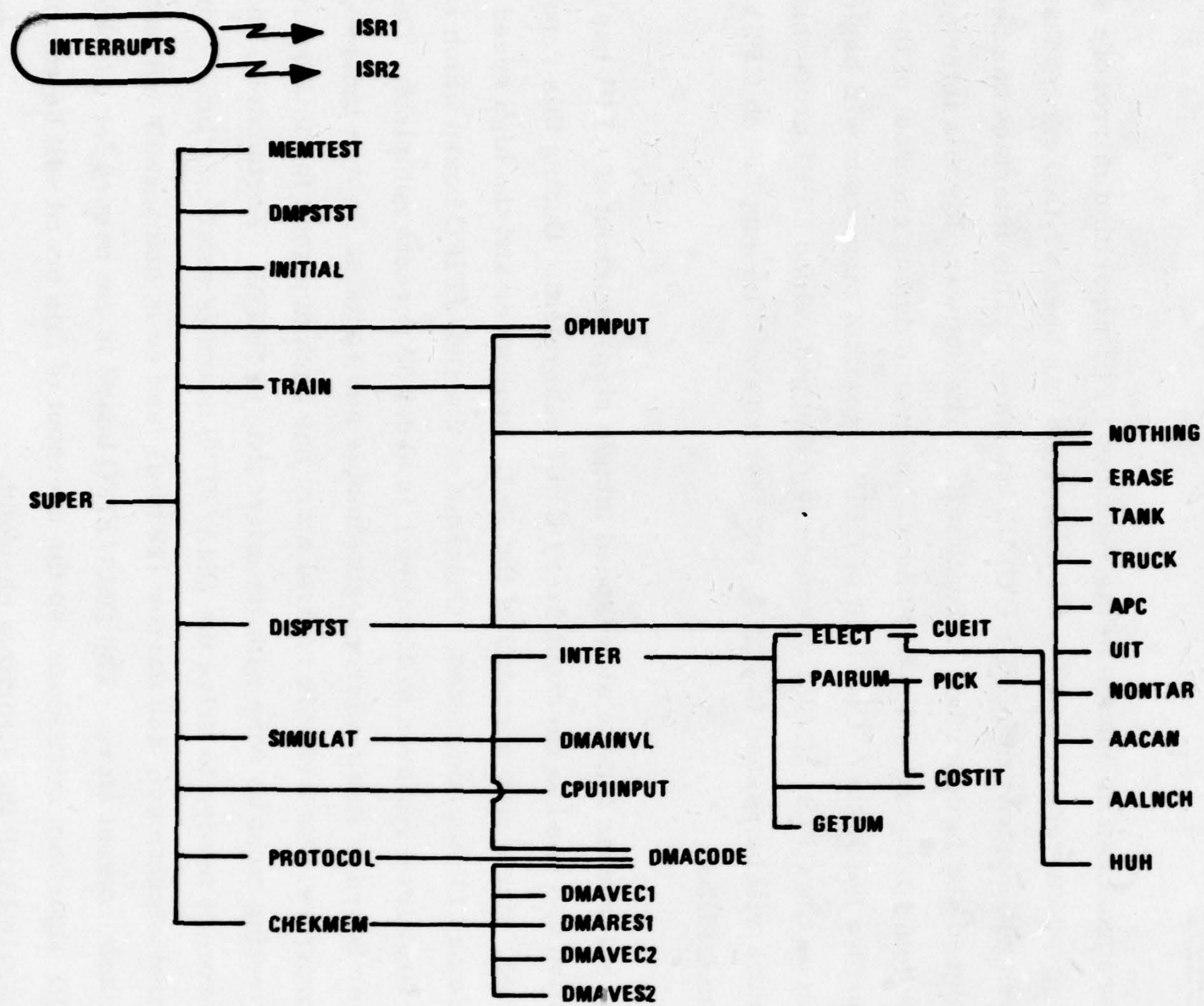


Figure 11. CPU2 Hierarchy of Software Modules

CPU1 SOFTWARE STATUS

Summary

There has been no change in the status of CPU1 algorithm microcode since the previous report; that is, all microcode has been written except that which will support the CPU1 - CPU2 interface. (This interface has been designed and is described in Section II.) The software for this interface is now being designed and coded and will be tested during checkout of the interface hardware. Checkout of CPU1 algorithm microcode will begin as soon as the CPU1 - CPU2 interface is debugged, since CPU1 processing results will be passed to CPU2, and test data will be supplied to CPU1 from CPU2.

Test microcode for the slow-speed (single step) checkout of CPU1 has been written and used to checkout every CPU1 subsystem. During this reporting period, single step checkout of the ALU, sequencer and the high speed RAM (Memory 1) was completed. Checkout of the DMA/FIFO board which transfers the first level features to Memory 1 is about 50 percent complete. The latter board is itself microprogrammable and contains its own independent microstore. No writable control store has been designed for this board so all testing must be done with its microcode in PROM. Fortunately, the microcode needed to drive the DMA/FIFO board is relatively brief (<50 microinstructions), and narrow (29 bits), and so is manageable without a writable control store. The DMA/FIFO board is not needed for checkout of CPU1 algorithm microcode, so the checkout of this board will be completed in parallel with the algorithm checkout.

DMA/FIFO Algorithm Description

The purpose of the algorithm implemented in microcode on the DMA/FIFO board is to transfer first level feature data from the internal generation output bus to CPU1 Memory 1. The input first-in first-out (FIFO) buffer and DMA controller hardware was originally described in Quarterly Report No. 4.[2] The hardware block diagram is shown in Figure 12 and the algorithm flowchart in Figure 13.

At the end of the active portion of each scan line, an "end of line" signal is produced by the interval generation hardware. If the DMA/FIFO microcontroller fails to sense end of line, it vectors to microcode, which checks to see if the CPU is trying to do I/O on the registers in the DMA address generator. Any pending I/O requests are then serviced, and the microprogram then re-loops to the end of line check.

If end of line is sensed, the microcontroller vectors to microcode, which will transfer to Memory 1 any feature data produced during the last scan line. This microcode first determines whether any feature data was produced by loading the feature word count into the word count register of the DMA address generator. If this count is zero, the address generator will immediately give a DONE signal. If DONE is sensed, the microcode re-loops to the end of line check.

If feature data is produced, each word is loaded into FIFO1. (FIFO1 and FIFO2 in the block diagram (Figure 12) are two sets of 64-word by 16-bit

[2] D. E. Soland, et al., "Prototype Automatic Target Screener," Quarterly Progress Report No. 4, Contract No. DAAK70-77-C-0248, Honeywell Systems and Research Center, Minneapolis, MN, Ocotober 20, 1978.

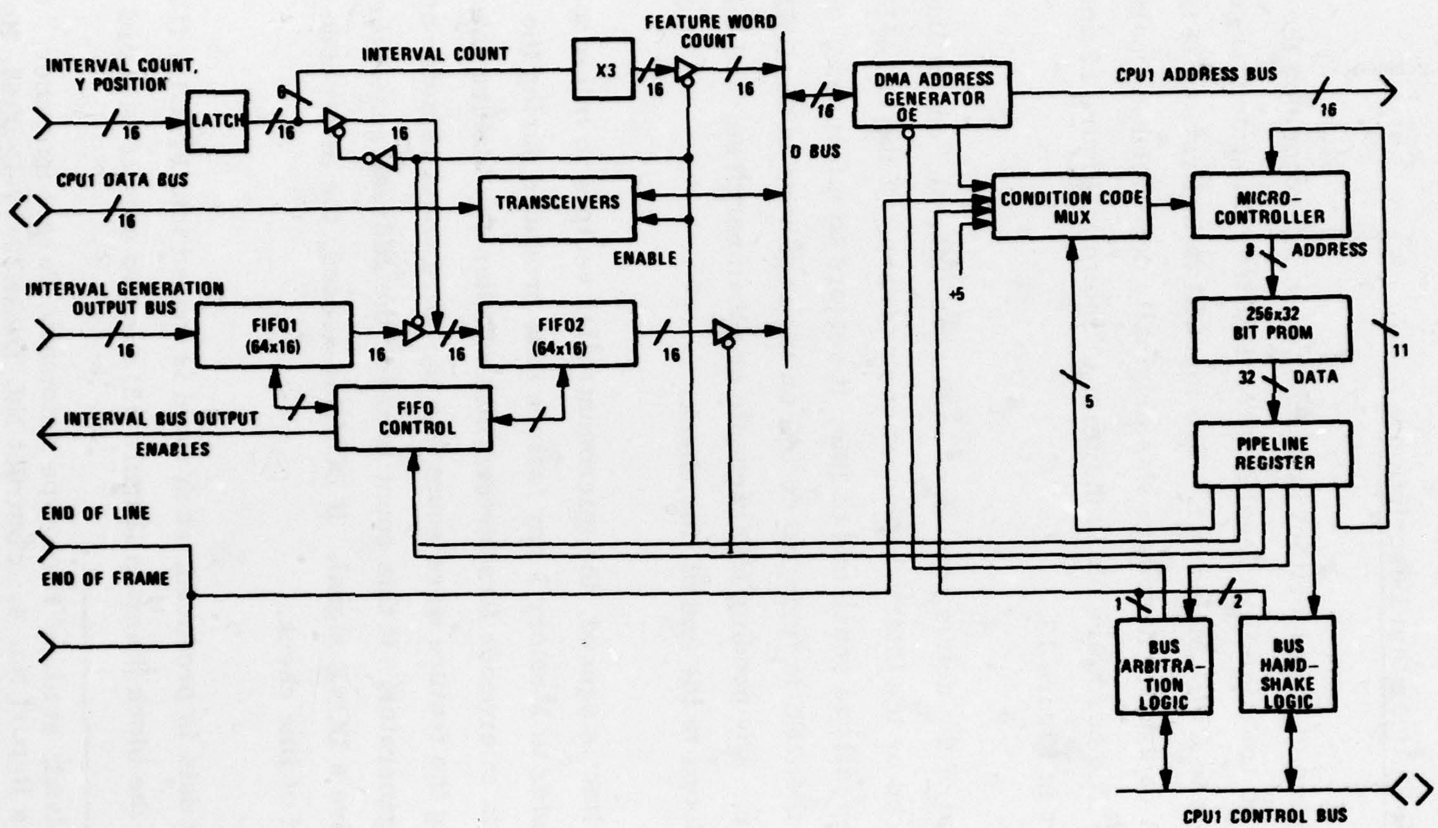


Figure 12. Block Diagram of DMA/FIFO

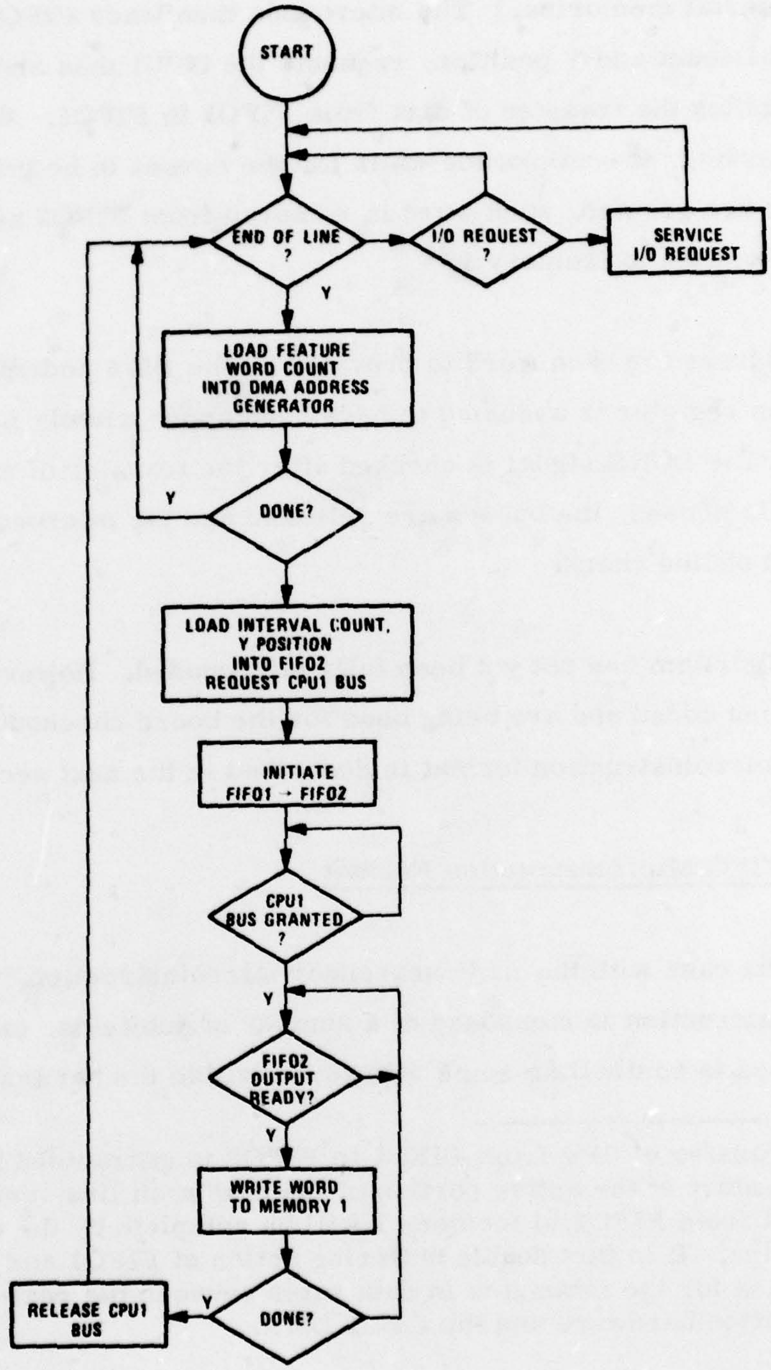


Figure 13. Flowchart of DMA/FIFO Algorithm

FIFO serial memories.) The microcode then loads FIFO2 with the packed interval count and Y position, requests the CPU1 data and address busses, and enables the transfer of data from FIFO1 to FIFO2. While this transfer is occurring, the microcode waits for the busses to be granted. Once the busses are granted, each word is removed from FIFO2 as it becomes available and is written to Memory 1.

The address for each word is provided by the DMA address generator, whose address register is assumed to have been appropriately initialized by the CPU. The DONE signal is checked after the transfer of each word. Once DONE is sensed, the busses are released and the microcode re-loops to the end of line check.*

This algorithm has not yet been fully microcoded. However, pieces of it have been coded and are being used for the board checkout. The DMA/FIFO microinstruction format is described in the next section.

DMA/FIFO Microinstruction Format

As is the case with the ALU/sequencer microinstruction, the DMA/FIFO microinstruction is composed of a number of subfields, each of which is dedicated to controlling some subsystem within the hardware.

*The transfer of data from FIFO1 to FIFO2 is guaranteed to be complete by the start of the active portion of the next scan line, and the transfer of data from FIFO2 to Memory 1 will be complete by the end of the next scan line. It is this double buffering action of FIFO1 and FIFO2 that compensates for the mismatch in data rates between the real-time interval generation hardware and the CPU1 DMA.

The microinstruction format is shown in Figure 14. The AMDASM micro-assembly language[3] is used to generate the bit pattern for each of the various subfields shown. These subfields have been grouped into two fields: one dedicated to microinstruction sequencing functions and the other encompassing all the remaining non-sequencing functions. Note that bit 16 is unused.

Field 1: Microsequencer Control (bits 0-15)--

Subfields:

1. Branch Address (bits 0-7)
   The microstore is 256 words deep; hence, the 8-bit branch address.

2. Branch Instruction (bits 8-10)
   The heart of the sequencer is a pair of bit slice microcontroller ICs; namely the Am 2909 and Am 2911. A set of eight instructions is implemented for them in 32x8-bit PROM. These instructions and their mnemonics are shown in Figure 15. These instructions are essentially a subset of those implemented on-chip in the Am 2910. Default: CONT.

3. MUX Polarity (bit 11)
   This and the next subfield control the pair of Am 2922s which are used to implement the condition code multiplexor. This bit controls the polarity of the MUX outputs. Mnemonics are shown in Figure 16. Default: NOINVERT

---

[3] Soland, et al., pp. 55, 56.

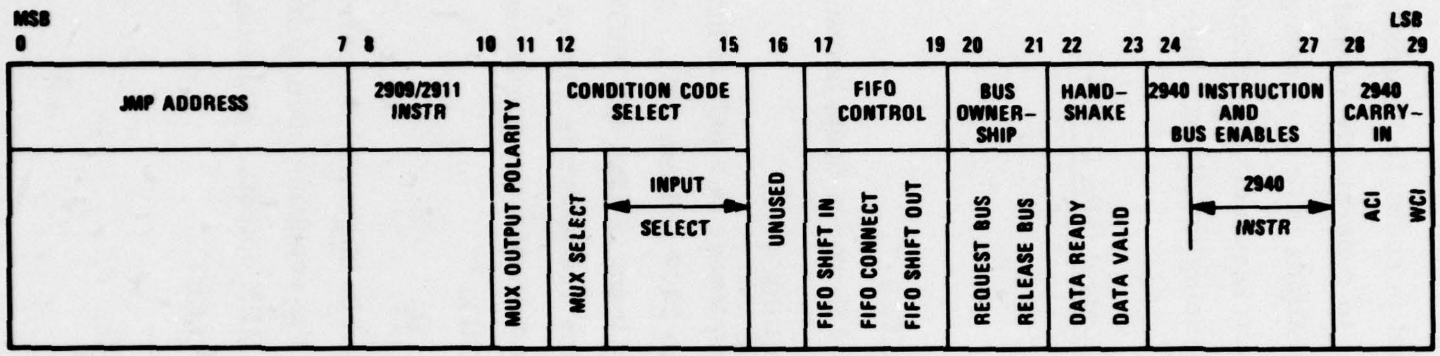


Figure 14. DMA/FIFO Microinstruction Format

```
JZ:      EQU Q#0   ;JUMP ZERO
CJS:     EQU Q#1   ;CONDITIONAL JUMP SUBROUTINE PIPELINE
CJP:     EQU Q#2   ;CONDITIONAL JUMP PL
CJV:     EQU Q#3   ;CONDITIONAL JUMP VECTOR
CRTN:    EQU Q#4   ;CONDITIONAL RETURN FROM SUBR
CJPP:    EQU Q#5   ;CONDITIONAL JUMP PL AND POP STACK
LOOP:    EQU Q#6   ;TEST END LOOP
CONT:    EQU Q#7   ;CONTINUE
```

Figure 15. Branch Instructions

```
INVERT:  EQU B#0    ;INVERT MUX OUTPUT
NOTINVERT:EQU B#1   ;NO INVERSION
```

Figure 16. Condition Code Polarity Selects

4. Condition Code Select (bits 12-15)

The various conditions and their mnemonics are shown in Figure 17. Not all of them are synchronized with the sequencer clock (that is, some are unlatched). Such conditions should be tested only when it is known they are stable. Default: TRUE.

```
TRUE:     EQU H#0    ;UNCONDITIONAL
DMADONE:  EQU H#1    ;DMA TRANSFER COMPLETE (UNLATCHED)
ORFIFO:   EQU H#2    ;FIFO2 OUTPUT READY
EOLINT:   EQU H#3    ;END OF LINE
EOFINT:   EQU H#4    ;END OF FRAME
DMAIO:    EQU H#5    ;I/O REQUEST ON DMA REGISTERS (UNLATCHED)
BUSGRNTD: EQU H#6    ;BUS GRANTED TO DMA
MEMREAD:  EQU H#7    ;MEMORY READ (UNLATCHED)
DATVALID: EQU H#8    ;DATA VALID
DATREADY: EQU H#9    ;DATA READY (UNLATCHED)
```

Figure 17. Condition Code Selects

Field 2: DMA and FIFO Control (bits 17-29)--

Subfields:

1. FIFO Control (bits 17-19)

   These bits control the two sets of FIFOs (FIFO1 and FIFO2) mentioned in the DMA/FIFO algorithm description. Mnemonics appear in Figure 18. Regardless of the instruction, data is always enabled into FIFO1 from the interval generation hardware as long as FIFO1 is not full. NOPFIFO disables the transfer of data from FIFO1 to FIFO2 and holds FIFO2 in its current state. SO enables the transfer of data between FIFO1 and FIFO2 and also enables the transfer of data out of FIFO2 to the data bus transceivers as this data becomes available. CONNECT enables the data transfer between two sets of FIFOs without shifting out FIFO2. SI loads the packed interval and Y position count into FIFO2. Default: NOPFIFO.

```
NOPFIFO: EQU Q#0    ;NO OPERATION
SO:      EQU Q#3    ;CONNECT FIFO'S AND SHIFT OUT FIFO2
CONNECT: EQU Q#2    ;CONNECT FIFO'S (NO SHIFT OUT)
SI:      EQU Q#4    ;SHIFT IN FIFO2
```

Figure 18. FIFO Control

2. Bus Arbitration (bits 20-21)

These bits allow the DMA to get control of the CPU1 data and address busses prior to data transfer and allow it to release the busses once the transfer is complete. Mnemonics are shown in Figure 19. Default: NCBUS (bus ownership is left in its current state).

```
NCBUS:   EQU B#01   ;NO CHANGE IN BUS
REQBUS:  EQU B#11   ;REQUEST BUS
RELBUS:  EQU B#00   ;RELEASE BUS
```

Figure 19. Bus Arbitration

3. Memory Handshake (bits 22-23)

These bits allow the DMA/FIFO board to use the data and address busses using the standard CPU1 bus protocols[4]. Menmonics are shown in Figure 20. Once DMA/FIFO has assumed bus ownership, it initiates data transfers by asserting DVALID (data valid) and

---

[4] Soland, et al., pp. 31, 32.

```
DVALID:   EQU B#01   ;SET DATA VALID
DRDY:     EQU B#10   ;SET DATA READY
CLEARHS:  EQU B#00   ;CLEAR DATA VALID AND DATA READY
```

Figure 20. Memory Handshake Control

waits for acknowledgement from the addressed device via the data ready signal (tested using the DATREADY mnemonic on the condition code MUX). When it is not bus master, DMA/FIFO responds to requests from the CPU by sensing data valid (via the DATVALID mnemonic) and asserting DRDY (data ready). Such requests are made whenever the CPU needs to read or write the registers in the DMA address generator on the DMA/FIFO board. Default: CLEARHS (sets data valid and data ready to their inactive states).

4. DMA Controller Instruction and Bus Enables (bits 24-27)
A pair of Am 2940s form the 16-bit address generator for the DMA. These bits, besides providing the instruction inputs to the Am 2940s, also control the enables for the internal data bus on the DMA/FIFO board. This is the bus labelled D in Figure 12. This bus has four transmitters, (the Am 2940s, the data bus, the output on FIFO2, and the feature word count) and two receivers (the Am 2940s and the data bus). The instruction mnemonics appear in Figure 21. These instructions encode D bus enables for the following transmitter-receiver pairs: AM 2940s - data bus (instruction RDCR, RDWC, RDAC), data bus - Am 2940s (instructions WRCR, LDAD, LDWCDB), FIFO2 output - data bus (instructions REINF, ENCTF), feature word count - data bus (instructions REINWCR, ENCTWCR), and

feature word count - AM 2940s (instruction LDWCWCR). Default: ENCTF (enables address and word counters on Am 2940s and enables FIFO2 output on to data bus).

```
WRCR:     EQU H#0 ;WRITE CONTROL REG AND ENABLE DATA BUS ONTO D
RDCR:     EQU H#1 ;READ CONTROL REG AND ENABLE D ONTO DATA BUS
RDWC:     EQU H#2 ;READ WORD COUNTER AND ENABLE D ONTO DATA BUS
RDAC:     EQU H#3 ;READ ADDRESS COUNTER AND ENABLE D ONTO DATA BUS
REINF:    EQU H#4 ;REINITIALIZE COUNTERS AND ENABLE FIFO2 OUTPUT ONTO DATA BUS
LDAD:     EQU H#5 ;LOAD ADDRESS AND ENABLE DATA BUS ONTO D
LDWCDB:   EQU H#6 ;LOAD WORD COUNT AND ENABLE DATA BUS ONTO D
ENCTF:    EQU H#7 ;ENABLE COUNTERS AND ENABLE FIFO2 OUTPUT ONTO DATA BUS
REINWCR:  EQU H#C ;REINITIALIZE COUNTERS AND ENABLE FEATURE WORD COUNT ONTO DATA
LDWCWCR:  EQU H#E ;LOAD WORD COUNT AND ENABLE FEATURE WORD COUNT ONTO D
ENCTWCR:  EQU H#F ;ENABLE COUNTERS AND ENABLE FEATURE WORD COUNT ONTO DATA BUS
```

Figure 21. DMA Controller Instruction and Bus Enables

5. DMA controller carry-in control (bits 28-29)

These bits are the complemented carry-ins to the address and word counters on the Am 2940s. Mnemonics are shown in Figure 22. Default: NCI.

```
CI:     EQU B#00   ;CARRY-IN
NCI:    EQU B#11   ;NO CARRY-IN
```

Figure 22. DMA Address Generator Carry-In

Summary--The mnemonics used to designate Field 1 and Field 2 are BRANCH and DMA, respectively. The AMDASM format DEFs for each of them is shown in Figure 23. Each DMA/FIFO microinstruction written in AMDASM must contain the BRANCH and DMA mnemonics; mnemonics for the various subfields are optional, and when omitted, the specified defaults take effect. An example of DMA/FIFO microcode appears in the next section.


```
BRANCH:   DEF 8VX,3V0#7,1VB#1,4VH#0,15X
DMA:      DEF 17X,3V0#0,2VB#01,2VB#00,4VH#7,2VB#11,2X
```

Figure 23. AMDASM Format Definitions

DMA/FIFO Microcode Example

The microcode shown in Figure 24 services CPU I/O requests on the registers in the DMA address generator and has been used in the DMA/FIFO board checkout.

The CJV branch instruction is used to vector to microcode needed to service a particular I/O request. The least significant three bits of the eight-bit vectored address are the inverted least significant three bits from the CPU memory address bus; the most significant four bits are set in microcode, and the remaining bit is the inverted memory read/write signal.

```
AMD AMDASM MICRO ASSEMBLER, V2.0
TEST MICROCODE FOR DMA/FIFO BOARD

          ;
          ; CHECK FOR DATA VALID
001C DMATEST: BRANCH DMATEST,CJP,INVERT,DATVALID & DMA
001C 0001110001001000 X0000100011111XX
          ;
          ; CHECK FOR I/O REQUEST FROM CPU1
001D          BRANCH DMATEST,CJP,INVERT,DMAIO & DMA
001D 0001110001000101 X0000100011111XX
          ;
          ; VECTOR TO APPROPRIATE ADDRESS
001E          BRANCH H#20,CJV,,TRUE & DMA
001E 0010000001110000 X0000100011111XX
001F          BRANCH & DMA
001F XXXXXXXX11110000 X0000100011111XX
          ;
          ; WRITE CONTROL REGISTER
0020          DMA ,,,WRCR & BRANCH READY,CJP,,TRUE
0020 0011000001010000 X0000100000011XX
0021          DMA ,,,WRCR & BRANCH READY,CJP,,TRUE
0021 0011000001010000 X0000100000011XX
          ;
          ; WRITE WORD COUNT
0022          DMA ,,,LDWCDB & BRANCH READY,CJP,,TRUE
0022 0011000001010000 X0000100011011XX
0023          DMA ,,,LDWCDB & BRANCH READY,CJP,,TRUE
0023 0011000001010000 X0000100011011XX
          ;
          ; WRITE ADDRESS COUNT
0024          DMA ,,,LDAD & BRANCH READY,CJP,,TRUE
0024 0011000001010000 X0000100010111XX
0025          DMA ,,,LDAD & BRANCH READY,CJP,,TRUE
0025 0011000001010000 X0000100010111XX
          ;
          ; UNUSED I/O ADDRESSES
0026          BRANCH DMATEST,CJP,,TRUE & DMA
0026 0001110001010000 X0000100011111XX
0027          BRANCH DMATEST,CJP,,TRUE & DMA
0027 0001110001010000 X0000100011111XX
          ;
          ; READ CONTROL REGISTER
0028          DMA ,,,RDCR & BRANCH READCR,CJP,,TRUE
0028 0011000101010000 X0000100000111XX
0029          DMA ,,,RDCR & BRANCH READCR,CJP,,TRUE
0029 0011000101010000 X0000100000111XX
          ;
```

Figure 24. Test Microcode for DMA/FIFO Board

```
      ; READ WORD COUNT
002A        DMA ,,,RDWC & BRANCH READWC,CJP,,TRUE
002A 0011001001010000 X0000100001011XX
002B        DMA ,,,RDWC & BRANCH READWC,CJP,,TRUE
002B 0011001001010000 X0000100001011XX
      ;
      ; READ ADDRESS COUNT
002C        DMA ,,,RDAC & BRANCH READAC,CJP,,TRUE
002C 0011001101010000 X0000100001111XX
002D        DMA ,,,RDAC & BRANCH READAC,CJP,,TRUE
002D 0011001101010000 X0000100001111XX
      ;
      ; UNUSED I/O ADDRESSES
002E        BRANCH DMATEST,CJP,,TRUE & DMA
002E 0001110001010000 X0000100011111XX
002F        BRANCH DMATEST,CJP,,TRUE & DMA
002F 0001110001010000 X0000100011111XX
      ;
      ; SIGNAL DATA READY FOR WRITE OPERATION
0030 READY: DMA ,,DRDY & BRANCH DMATEST,CJP,,TRUE
0030 0001110001010000 X0000110011111XX
      ;
      ; SIGNAL DATA READY FOR CONTROL REGISTER READ
0031 READCR: DMA ,,DRDY,RDCR & BRANCH DMATEST,CJP,,TRUE
0031 0001110001010000 X0000110000111XX
      ;
      ; SIGNAL DATA READY FOR WORD COUNT READ
0032 READWC: DMA ,,DRDY,RDWC & BRANCH DMATEST,CJP,,TRUE
0032 0001110001010000 X0000110001011XX
      ;
      ; SIGNAL DATA READY FOR ADDRESS COUNT READ
0033 READAC: DMA ,,DRDY,RDAC & BRANCH DMATEST,CJP,,TRUE
0033 0001110001010000 X0000110001111XX
      ;
      ;
      END

TOTAL PHASE 2 ERRORS =    0
```

Figure 24. Test Microcode for DMA/FIFO Board (concluded)

SECTION IV

PLANS FOR THE NEXT REPORTING PERIOD

During the early portion of the next reporting period, CPU1 checkout will be completed and microcode checkout will begin. The software required for training will also be checked out once the interface between CPU1 and CPU2 is checked out.

The interval boards will be checked out and system integration will be done. The power supplies will be incorporated into the ATR once they are received. This should complete the PATS hardware checkout.

A preliminary test plan will be written during the next reporting period and submitted for discussion and approval.